REMARKS/ARGUMENTS

These remarks are in response to the Office Action dated May 24, 2004. Claims 1-21 are pending in the present application.

Objections to the Specification

The Examiner objected to the Abstract and the title of the present invention. Applicants have amended the title to providing a better description of the invention, and have amended the Abstract to conform with the MPEP §608.01(b). No new matter has been added.

Objections to the Drawings

The Examiner objected to the drawings under 37 CFR 1.83(a) in that every feature of the invention specified in the claims was not shown in the drawings. In response, Applicants have submitted new drawings, FIG. 2 and FIG. 3. FIG. 2 is a flowchart illustrating a method for optimizing command execution in a database system according to a preferred embodiment of the present invention. FIG. 3 is a flowchart illustrating a process for verifying the accuracy of the selected row value according to a preferred embodiment of the present invention. The Specification has been amended to incorporate each flowchart. No new matter has been entered.

Claim Rejections

In the Office Action, the Examiner rejected claims 1-21 under 35 U.S.C. §102(a) as being unpatentable over Josten et al. (U.S. Patent No. 5,574,902) in view of Ponnekanti (U.S. Patent No. 6,591,269). In so doing, the Examiner stated:

> Regarding claim 1, Josten teaches a method for optimizing command execution in a database system, wherein data records are stored on a plurality of data pages therein (col. 5, line 65-col. 6, line 6), the method comprising the steps of:
>
> (a) 'providing an identifier to each data page' as an ordinal number (ORD#) is assigned to a data page buffer control block (BCB in dirty page list (DPL) (col. 7, lines 10-16 and 42-52; col. 11, lines 44-46, and Fig. 2, element 44 (i.e., ordinal #));
>
> (b) 'selecting a data record from a data page' as list 42 includes a series of the ordinal number from DPL that are associated with data pages in the

LCB that were accessed by the transaction corresponding to TPL-1 (col. 7, line 53 – col. 8, line 2; col. 5, lines 5-10; col. 7, lines 10-16; col. 8, lines 1-7; col. 8, lines 46-51);

(c) 'copying the selected data record to a second storage area' as the data manager issues a SETWRITE request to indicate intent to update the named data page (col. 7, lines 10-16; col. 8, lines 9-18);

(d) 'verifying that the selected data record has not been modified since the time that it was copied to the second storage area based upon the identifier' as a test is made to detect a consecutive update to the same data page by comparing the ORD# of the last entry of the transaction page list (TPL) with the ORD# of the new entry in the buffer control block (col. 11, lines 28-38); and

(e) 'executing the command' as committing transactions schedules write I/Os for all TPL entries (col. 18, lines 3-7).

a). Josten does not explicitly teach the identifier indicating when any of the data records contained therein were last modified.

Ponnekanti, however, teaches the log records contain only the PAGEIDs and the timestamps of the source page and the target page and the positions of the first and last key that were copied (col. 11, lines 47-49).

It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of the cited references because Ponnekanti's teaching would have allowed Josten's to enhance the speed in which the database server stores, retrieves, and processes particular data records as indicated by Ponnekanti at col. 3, lines 1-10 and col. 19, lines 40-50.

Applicants respectfully disagree.

The present invention is directed to optimizing command execution in a relational database management system RDBMS where data records are stored in data pages. According to the present invention, *each* data page is provided with an identifier which indicates when the page was modified last. Thus, whenever a record on the data page is modified, the identifier necessarily changes. When the RDBMS receives a command to access a data record on a data page, such as an UPDATE or DELETE command, the system copies the selected *data record* from the page, along with the page's identifier, and stores them temporarily in a secondary storage area.

The system uses the stored identifier to verify that the copied record has not been modified during the period in which it was copied and stored. For instance, the system compares the stored identifier with a current identifier for the page. If the page has not been modified, the

stored identifier and the current identifier will match, and the system can conclude that the stored data record matches the current record in the database. Accordingly, when the identifiers match, the system is not required to compare the data record in the temporary copy with the record in the current table, which can be a costly event if the records are extensive.

By providing each data page with the identifier and then storing the identifier with the data record in the secondary storage, the RDBMS can quickly and easily determine whether the data page containing the stored data record was modified by comparing identifiers. Accordingly, the present invention minimizes the number of times the RDBMS must go into the data page itself to verify that the stored record is consistent with the current record.

In contrast, Josten is directed to a method for destaging updated locally cache pages in a multisystem shared disk environment, such as a Sysplex, with a high-speed shared electronic store. In Josten, each computer processing complex (CPC) caches data pages in its local cache buffer (LCB). When one of those pages is updated, it eventually must be written to shared stable storage ("externalized") and cross-invalidated in other CPCs when a transaction commits. (Col. 4, lines 8-13). Josten presents (a) an efficient process for identifying the set of cached pages that must be externalized and (b) a fast process for scheduling of the externalizing write I/Os so that the corresponding transaction can release its locks. (Col. 4, lines 13-18). To do this, Josten assigns an ordinal number (ORD#) to a dirty data page, which is defined as a page that contains a record that has been updated by a transaction, but has not yet been written to external stable storage. Josten then places the ORD# in a list associated with the transaction (transaction page list (TPL)). (Col. 7, lines 1-18).

Ponnekanti is directed to performing an online rebuild of a B+ tree index by copying the index rows to newly allocated pages in the key order so that good space utilization and clustering are achieved. (Abstract). In Ponnekanti, index rows are copied to newly allocated pages in the

key order, and old pages are deallocated. Multiple leaf pages are rebuilt and then changes to higher levels are propagated. While propagating leaf level changes to higher levels, level 1 pages are reorganized. (Abstract). Thus, the need for a separate pass is eliminated.

Applicants respectfully submit that Josten and Ponnekanti fail to teach or suggest "providing an identifier to *each* data page" in a database system, as recited in claims 1 and 9, and "means for providing an identifier on *each* data page" in the database system, as recited in claim 17, where the identifier indicates "when any of the data records contained therein were last modified."

According to the Office Action, the Examiner suggests that the identifier of the present invention is taught by Josten's ORD# which is assigned to a data page in the local cache buffer (LCB) when its status changes from clean to dirty. (Col. 7, lines 10-16). Thus, a data page that remains clean is not assigned an ORD#. In the present invention, the identifier is provided to *each data page*, and not just to those that contain an updated data record. Accordingly, Applicants respectfully submit that Josten's ORD# is not equivalent to the identifier of the present invention.

Moreover, the identifier indicates *when* any of the data records contained in the data page was modified last. The Examiner concedes that the ORD# does not provide such an indication, but suggests that Ponnekanti's log records do at column 11, lines 47-49. Applicants respectfully disagree. In Ponnekanti, the application runs as a sequence of transactions, with each transaction rebuilding leaf pages in the page chain. At the end of each transaction, the new pages generated are flushed to disk and then the old pages that were removed from the tree are made available for fresh allocations. The log records contain PAGEIDs and timestamps of the source page and the target page, as well as positions of a first and a last key copied. (Column 11, lines 30-49).

The PAGEID and timestamp of a source or target page has no relation to Josten's ORD#. In addition, nothing in Ponnekanti teaches or suggests that the timestamp of the source or target page indicates "when any of the data records contained therein were last modified," as recited in claims 1, 9 and 17. Accordingly, Applicants respectfully submit that the combination of Josten and Ponnetanki fail to teach or suggest the identifier of the present invention, as recited in claims 1, 9 and 17.

Furthermore, neither reference teaches or suggests "copying *the selected data record* to a second storage area," as recited in claims 1, 9 and 17. In the present invention, a data record is selected from a data page in the database, and then copied to a second storage area so that access to the database is not necessarily required to verify the accuracy of the data record. In contrast to the present invention, Josten updates a data page by retrieving the *data page*, not just the data record, from the database and *upstaging the data page* to the cache buffer pool. A SETWRITE request is then issued to indicate an intent to update the named data page. (Col. 8, lines 9-18). Josten discloses upstaging the *data page* to the cache buffer pool from the database. Nothing in Josten or Ponnekanti teaches or suggests selecting the data record from the data page, and then "copying the selected data *record* to a second storage area," as recited in claims 1, 9 and 17.

Finally, Applicants respectfully submit that neither reference teaches or suggests "verifying that the selected data record has not been modified since the time that it was copied to the second storage area *based upon the identifier*," as recited in claims 1 and 9, and "means for verifying that the selected data record has not been modified since the time that it was copied to the second storage area by determining that the stored identifier is the same as the current identifier from the data page," as recited in claim 17. In Josten, the ORD# for a dirty page does not change once the page becomes dirty. If a transaction updates a data page *more than once*, the ORD# for the dirty data page does not change. Thus, in Josten, it is not possible to verify that

"the selected data record has not been modified since the time that it was copied to the second storage area *based upon the identifier*," as recited in claims 1 and 9, and Josten does not teach or suggest "means for verifying that the selected data record has not been modified since the time that it was copied to the second storage area by determining that the stored identifier is the same as the current identifier from the data page," as recited in claim 17.

In the Office Action, the Examiner states that this feature is taught by Josten at column 11, lines 28-38. Applicants respectfully disagree. In the cited portion, Josten discusses avoiding duplicate TPL entries by determining whether a consecutive update has been performed on the same data page by comparing the ORD# of the last entry of the TPL with the ORD# of the new entry. If the ORD#s are *different*, the new entry is made at the end of the TPL. This process does not verify "that the selected data record has not been modified since the time that it was copied to the second storage area," as recited in claims 1, 9 and 17.

For the foregoing reasons, Applicants respectfully submit that claims 1, 9 and 17 are allowable over Josten and Ponnekanti. Claims 2-8, 10-16, and 18-21 depend on claims 1, 9 and 17, respectively, and therefore, the above arguments apply with full force. Thus, claims 2-8, 10-16, and 18-21 are also allowable over Josten in view of Ponnekanti.

Conclusion

In view of the foregoing, Applicants submit that claims 1-21 are allowable over the cited references. Applicants respectfully request reconsideration and allowance of the claims as now presented.

Applicants' attorney believes that this application is in condition for allowance. Should any unresolved issues remain, Examiner is invited to call Applicant's attorney at the telephone number indicated below.

Respectfully submitted,
SAWYER LAW GROUP LLP

August 24, 2004
Date

Joseph A. Sawyer, Jr.
Attorney for Applicant(s)
Reg. No. 30,801
(650) 493-4540